A Path to Oscillation Free Controller Changes*

Milton Cunguara, Tomás Oliveira e Silva
DETI/IEETA/University of Aveiro
Campus Santiago, 3810-193
Aveiro, Portugal
{milton.cunguara,tos}@ua.pt

Paulo Pedreiras
DETI/IT/University of Aveiro
Campus Santiago, 3810-193
Aveiro, Portugal
pbrp@ua.pt

Abstract

Driven by the need to make a more efficient use of communication and computational resources, recently, a strong research effort has been devoted to the study of control systems integrating several controllers, with the goal of selecting the controller that allows to meet the desired quality of control while minimizing resource usage. However, most of the research made so far has been focused on the rules for controller change, neglecting the full impact of controller changes, namely the resulting oscillations. Consequently, in most such systems, controller changes may cause output oscillations that can compromise the gains achieved by the use of multiple controllers. In this paper, first the cause for oscillations in the presence of period changes is put forward, then a solution based in a change of basis matrix is presented. A simulation illustrate the feasibility of the proposed methodology.

1 Introduction and Motivation

Currently, almost all control systems are executed in digital micro-processors. Such micro-processors allow for a more flexible controller design, since a single micro-processors can execute more than one process and the software used in one application can easily be ported to other micro-processors, as opposed to analog compensators which hardly can be reused. This motive, plus the ever falling price of micro-processors, their ever increasing performance and the scalability achieved the use of fieldbuses, made the use of micro-processors the *de facto* option in control applications.

Due to this flexibility, such micro-processors often execute a number of processes that compete for its computational resources. Designing systems according to worst-case requirements may lead to expensive and inefficient designs. Thus, to save system resources, a number of approaches were put forward in the literature (see section 2). Many of them rely on controller changes: using less resource (CPU and network) demanding, albeit less efficient, controllers whenever the system has a small error, and more resource demanding and efficient controllers when the system has a large error.

However, in hitherto proposed controller change methodologies, controller transitions are generally followed by an output oscillation. At best, these oscillations degrade the quality of control. However, it may happen that such oscillations in turn cause another controller change, that itself triggers another oscillation, so on and so forth. Consequently, systems with multiple controllers tend to oscillate between two or more controllers, even when the system conditions are not changing, thus, defeating the purpose of controller change. Hence, for ensuring smooth and oscillations-free transitions it is germane to manage controller changes in a way that eliminates output oscillations, while preserving an efficient resource utilization.

The remaining of this paper is organized as follows: Section 2 presents a review of the related work, focusing in network-control co-design. Section 3 introduces the problems that arise when a period switch is performed both formally and informally (through an example), followed by a solution. Section 4 presents an evaluation of the proposed solutions. Finally, in Section 5 some conclusions are drawn and future lines of inquiry presented.

2 Related Work

This section provides an overview of scientific contributions that are close and relevant to the ones covered in this paper.

[1] proposed the use of several controllers, each tuned to a given period. The scheduler would choose a controller with a given period according to the error. This idea was somewhat complemented in [2], with a study of optimal transition error levels, though the former study was centered around distributed systems whereas the latter was focused on centralized systems. Further work includes the investigation of when a number of pre-calculated controllers outperforms dynamical chosen ones.

A number of studies, for example [3] and [4], discuss whether the controller change should be based in instantaneous or in filtered/accumulated measures of error. [3]

*This work was partially supported by the Portuguese Government through FCT - "Fundação para a Ciência e a Tecnologia" in the scope of project Serv-CPS -PTDC/EEA-AUT/122362/2010 and Ph.D. grant - SFRH/BD/62178/2009/J53651387T4E

favors instantaneous measures, because the advantages do not compensate the extra computational burden. [4] presents similar experiments, though with different systems and concludes that filtering is worthwhile.

In [5] and [6] the authors present the co-design approach, which is (re)introduced with a new set of goals and paradigms. Each process has a given weight and the goal of the scheduler is to assign the periods in a way that minimizes the sum of the weights times the squared errors. [7] presents a different metric for minimization, a metric dubbed quality of control. Nevertheless, not only the mechanism is somewhat similar to its predecessors, but it may also present itself more difficult to minimize.

Using the CAN protocol, [4] discusses the advantages of period switch at network level, thus approaching a co-design perspective. In [8] and references therein, a discussion regarding feedback scheduling is presented. Feedback scheduling can be loosely defined as a scheduling methodology in which the task's current characteristics (e.g. period, worst-case execution time) are set based in current operating conditions. The problem of scheduling such processes has also made significant advances, one of the most relevant being the elastic task model [9], which draws from an analogy of strained springs. In this method the tasks utilization are adjusted to have a given utilization level. Each task has an individual weight that controls its relative level of adaptivity. This model is general enough to be ported to control situation without major modifications. Nonetheless, it was further generalized in [10] for encompassing other types of minimization in its task stretching. Nevertheless, all such studies assume the existence of a mechanism for correctly/optimally choose the weights of tasks. However, to the best of our knowledge, there is no study that provides a step in this direction.

[11] introduced non-periodic controllers in which the next activation instant is chosen based on current state and error. This approach has a few drawbacks, such as: 1) is inherently local, i.e. the controller must have a new sample in a moment that is unavailable in distributed systems. 2) it is not clear that there is no periodic controller that achieves the same quality of control. 3) its erratic activation pattern, that was further discussed in [12], renders its associated schedulers far more complex.

3 The proposal

Period change is (probably) the most commonly used controller change technique, hence this article will focus in this approach. In this type of controller change, the sample and actuation period of the controller are modified, accompanied by a change in the parameters.

3.1 Period Switch

In their operation, controllers store a number of linear combinations of past inputs/outputs. Upon a controller change the output almost certainly oscillate because after that change the state still *points* to the output in the instants induced by the previous period, as opposed to the instants induced by the current period.

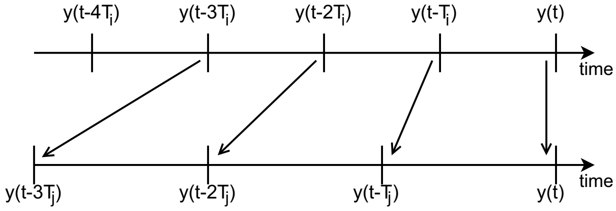


Figure 1: State in a classical controller change.

To shed some light into the problem, consider a controller with a period $\mathrm{T_i}$ and N state variables. At a given instant t there is a period change and the new period is $\mathrm{T_j}$. In the former case the state includes estimates of $[\mathrm{y(t)\ y(t-T_i)\ \cdots\ y(t-(N-1)T_i)}]$. However, when the transition occurs, the state does not automatically change to point to $[\mathrm{y(t)\ y(t-T_j)\ \cdots\ y(t-(N-1)T_j)}]$ (see Fig. 1), i.e. it continues to point into the same time instants. Since the controller is tuned for the former parameters it will produce suboptimal control values until the old state values are all replaced by new values spaced apart by $\mathrm{T_j}$. This *re-adaptation period* lasts for $\mathrm{N-1}$ samples. It is also elucidative to note that, for example, a controller that saves the values of the last N positions will oscillate whenever the controller changes, whereas a controller that saves the position, and its $\mathrm{N-1}$ successive derivatives does not oscillate upon controller change, since at the commutation time the state already points to the right place.

3.2 Problem Formulation

Consider a system SYS that is controlled using more than one controller at different sampling rates. Consider that the $\mathrm{i^{th}}$ controller is sampled at rate $\mathrm{T_i}$ and has an associated state-space representation $\mathrm{SYS_i}$[1], in which x is the state, u the input, y the output, k the sample order, A the state transition matrix, B the input matrix and C the output matrix.

$$x_i(k+1) = A_i x_i(k) + B_i u(k) \tag{1a}$$
$$y(k) = C_i x_i(k) \tag{1b}$$

and that the $\mathrm{j^{th}}$ controller has a sampling rate $\mathrm{T_j}$ and associated representation $\mathrm{SYS_j}$

$$x_j(k+1) = A_j x_j(k) + B_j u(k) \tag{2a}$$
$$y(k) = C_j x_j(k) \tag{2b}$$

Then, the goal is to devise a mechanism to switch from $\mathrm{SYS_i}$ to $\mathrm{SYS_j}$ without oscillation, i.e. find $\mathrm{x_j}$ in the space of $\mathrm{SYS_j}$ that corresponds to $\mathrm{x_i}$ in $\mathrm{SYS_i}$.

3.3 Proposed Solution

To achieve the goal sought for in the last subsection, lets introduce an auxiliary representation $\mathrm{SYS_{ja}}$, which is

[1] many controllers can easily be written in such way

sampled at rate T_j but its continuous time version has a state equal to the state of SYS_i, i.e let SYS_{ja} be

$$x_{ja}(k+1) = A_{ja}x_{ja}(k) + B_{ja}u(k) \quad (3a)$$
$$y(k) = C_i x_{ja}(k) \quad (3b)$$

The introduction of SYS_{ja} turns the original problem into a simpler one, i.e. a change of basis problem. Since, $x_i = x_{ja}$ by design, and both SYS_{ja} and SYS_j are sampled at same rate there should be a matrix P such that $x_j = Px_{ja}$. To find P, recall the well known similarity equation $A_{ja} = P^{-1}A_jP$, or

$$PA_{ja} - A_jP = 0 \quad (4)$$

This is the well known Sylvester equation [2]. Nonetheless, classical efficient approaches, e.g. [13] [14], fail to give satisfactory solutions to this particular problem. Instead they always provide the trivial solutions ($X = 0$ or $P = 0$ in this case). It is noteworthy that this equation also appeared in robust pole placement techniques as attested by [15] and references therein[3], however the mechanisms used to solve it are not easily portable to this problem. To solve this problem a less efficient mechanism was employed: the Kroenecker product approach. In this approach (4) is transformed into:

$$\left(I_n \otimes A_{ja} - A_j^T \otimes I_n\right) vec(P) = 0 \quad (5)$$

where $\mathrm{vec}(P)$ is a vectorized version of the matrix P and I_n is the $(n \times n)$ identity matrix.

Equation (5) is an eigen problem associated with the eigen value 0. Once the eigen problem is solved there will be a series of vectors that, when passed back into the initial matrix space (the inverse of vec), will give rise to a series of eigen matrices and any linear combination of such matrices is also a solution of the problem.

Nevertheless, this problem may have more than one solution. It is well known that the eigen values of $\left(I_n \otimes A - B^T \otimes I_n\right)$ are $\lambda_i(A) - \lambda_j(B)$, where $\lambda_i(A)$ is the i^{th} eigen value of matrix A. In this particular case A and B have the same eigen values, hence each eigen value of A produces m_i^2 zero eigen values in the matrix P, with m_i the multiplicity of the i^{th} eigen value. Thus, P has an eigen value of 0 with multiplicity $\sum_{i=1}^{q} m_i^2 \geq n$, where q is the number of distinct eigen values of A.

The eigen matrices presented above relate to the spectral decomposition of the matrix in question in the sense that they expand the matrix of eigen vectors (upon proper multiplication) in a sum-like fashion. Thus, in this framework, the extra eigen matrices are induced by the well known rank of the eigen space spawned by eigen values with high multiplicity.

Due to the similarity transformation, matrix P must verify

$$B_j = PB_{ja} \quad (6)$$

[2] $AX + XB = C$ in which $A = -B$ and $C = 0$.

[3] this mechanism is used in Matlab® function place

Algorithm 1 Algorithm for finding the change of basis matrix

```
K ← (I_n⊗A_ja−A_j^T⊗I_n)
ns ← nullspace(K)
for all Columns of ns do
    S_i ← vec(inv_vec(ns_i)B_ja)
end for
Ω ← S^−1 vec(B_j)
P ← ns×Ω
```

Equations (4) (or (5)) and (6) define matrix P. More precisely, the first equation defines a rank three tensor, that is conveniently written as a number of eigen matrices and the product of such matrix by the matrix B_{ja} can be written as:

$$PB_{ja} = \left(\sum_i \omega_i P_i\right) B_{ja} \quad (7)$$

were P_i are the eigen matrices and ω_i are unknown coefficients. Define now, $s_i \equiv vec(P_iB_{ja})$. Then (7) and (6) can be rewritten as

$$vec(B_j) = [s_1\ s_2 \cdots s_z]\,[\omega_1\ \omega_2 \cdots \omega_z]^T \quad (8)$$

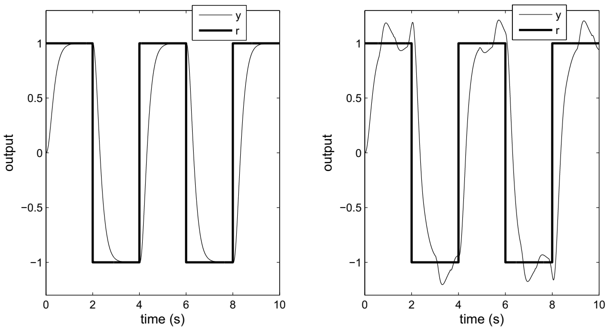
or simply, $vec(B_j) = S\Omega$ (were S and Ω are implicitly defined). The previous equation provides the values of ω that give the actual solution, $P = \sum_i \omega_i P_i$. Algorithm 1 summarizes the procedure that determines the change of basis matrix. The operation *inv_vec* used in the algorithm is the inverse of *vec* operator that transform matrices into row vectors. S^{-1} may not be exist, as discussed above, nonetheless, a proper pseudo-inverse may be used.

Obviously, if P switches from SYS_j to SYS_i, then P^{-1} switches from SYS_i to SYS_j. Also, since B is a matrix of size $(n \times r)$, equation (6) may be under-determined, meaning the initial problem has more than one solution.

4 Evaluation

This section intends to evaluate the main proposal of this paper. The test systems is represented using the *variable phase* (the state transition matrix is a *companion* matrix and the input matrix has one single 1 and $n-1$ zeros). This choice was made because it is one of the most frequently used in control practice, mostly due its ease of obtainability from physical characteristics of the systems.

The simulations were made using a square wave as reference signal. This wave changed between ± 1 with a duty cycle of 50% and a period of 4s. The controllers used standard regulators theory to track the reference signal. Pole-placement was used to place the poles regularly spaced in the interval $[0.85\,0.9]$. The simulations last for 10s. The controller was changed periodically with a period of 600ms. The subparts of the control system (i.e. sensor, controller, actuator) communicate through a network with a delay of 2ms and a jitter with a Poisson distribution with mean arrival time of also 2ms. The system commutated



(a) proposed (b) classical

Figure 2: Response of test system to a square-wave

between a sampling period of 16ms and 20ms, with the representation

$$x(k+1) = \begin{bmatrix} 0 & 1.0000 & 0 \\ 0 & 0 & 1.0000 \\ 0.3150 & -1.4300 & 2.1000 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(k)$$
$$y(k) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} x(k)$$

at 16ms and at 20ms it had the representation

$$x(k+1) = \begin{bmatrix} 0 & 1.0000 & 0 \\ 0 & 0 & 1.0000 \\ 0.2360 & -1.1990 & 1.9373 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(k)$$
$$y(k) = \begin{bmatrix} -0.0172 & 0.3045 & 1.4277 \end{bmatrix} x(k)$$

Figure 2 shows a simulation of this system. The classical approach caused the output to oscillate, whereas the proposed approach does not. The proposed approach had an Integral Square Error (ISE) of 1584 whereas the classical approach had an ISE of 1633. The small difference is due to the fact that the ISE was computed over the whole simulation. If a corrected ISE is used, i.e. computed only were the desired output is stable then the proposed approach has an ISE of 1.74×10^{-6} and the classical approach has an ISE of 14.19.

5 Conclusion

In this paper, the main reason for output oscillations after a controller change has been discussed. Summarizing, state variables before and after the controller change are not in agreement. A novel mechanism has been presented that finds a change of basis matrix that converts a state variable from one representation into another. Simulations of a distributed system, subject to network contention, were carried out with and without such mechanism. Results are in perfect agreement with the presented theory. Future contributions will address the algorithms efficiency as well as other types of controller changes, e.g. complexity changes.

References

[1] A. Antunes, P. Pedreiras, and A. Mota, “Adapting the sampling period of a real-time adaptive distributed controller to the bus load”, in *10th IEEE Conference on Emerging Technologies and Factory Automation, ETFA*, volume 1, sep 2005, pp. 1084–1087.

[2] A. Cervin, M. Velasco, P. Martí, and A. Camacho, “Optimal Online Sampling Period Assignment: Theory and Experiments”, in *IEEE Transactions on Control Systems Technology*, volume 19, 2010, pp. 1–9.

[3] R. Castane, P. Marti, M. Velasco, and A. Cervin, “Resource Management for Control Tasks Based on the Transient Dynamics of Closed-Loop Systems”, in *Proceedings of the 18th Euromicro Conference on Real-Time Systems*, 2006, pp. 171–182.

[4] A. Anta and P. Tabuada, “On the Benefits of Relaxing the Periodicity Assumption for Networked Control Systems over CAN”, in *Proceedings of the 2009 30th IEEE Real-Time Systems Symposium*, RTSS 09, Dec 2009, pp. 3–12, Washington, DC, USA. IEEE Computer Society.

[5] A. Cervin and J. Eker, “Feedback scheduling of control tasks”, in *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 5, 2000, pp. 4871–4876.

[6] A. Cervin, J. Eker, B. Bernhardsson, and K.-E. Årzén, “FeedbackFeedforward Scheduling of Control Tasks”, *Real-Time Syst.*, vol. 23, pp. 25–53, Jul 2002.

[7] G. Buttazzo, M. Velasco, and P. Marti, “Quality-of-Control Management in Overloaded Real-Time Systems”, *IEEE Trans. Comput.*, vol. 56, pp. 253–266, Feb 2007.

[8] C. Lozoya, P. Martí, M. Velasco, and J. M. Fuertes, “Control Performance Evaluation of Selected Methods of Feedback Scheduling of Real-time Control Tasks”, in *17th IFAC World Congress*, July 2008.

[9] G. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni, “Elastic scheduling for flexible workload management”, *Computers, IEEE Transactions on*, vol. 51, no. 3, pp. 289–302, mar 2002.

[10] T. Chantem, X. S. Hu, and M. Lemmon, “Generalized Elastic Scheduling”, in *Real-Time Systems Symposium, 2006. RTSS ’06. 27th IEEE International*, dec 2006, pp. 236–245.

[11] M. Velasco, P. Martí, J. M. Fuertes, C. Lozoya, and S. A. Brandt, “Experimental evaluation of slack management in real-time control systems: Coordinated vs. self-triggered approach”, *System Architecture*, vol. 56, pp. 63–74, Jan 2010.

[12] P. Martí, M. Velasco, and E. Bini, “The Optimal Boundary and Regulator Design Problem for Event-Driven Controllers”, in *Proceedings of the 12th International Conference on Hybrid Systems: Computation and Control*, HSCC ’09, 2009, pp. 441–444.

[13] R. H. Bartels and G. W. Stewart, “Solution of the matrix equation AX + XB = C [F4]”, *Commun. ACM*, vol. 15, no. 9, pp. 820–826, Sep 1972.

[14] G. Golub, S. Nash, and C. Van Loan, “A Hessenberg-Schur method for the problem AX + XB = C”, *Automatic Control, IEEE Transactions on*, vol. 24, no. 6, pp. 909–913, dec 1979.

[15] A. Varga, “Robust pole assignment techniques via state feedback”, in *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 5, 2000, pp. 4655–4660.